STANDARDS FOR ENCODING LINGUISTIC DATA

Martin Kay*

The RAND Corporation, Santa Monica, California

For every person who has successfully completed a linguistic or literary project using a computer, there are probably three or four who have embarked on such projects with the enthusiasm of their successful colleagues and the promises of a bright, young graduate student who has just learnt to program fresh in their minds and who, after a few frustrating weeks, months, or even years, abandon the project in disgust. Linguistic and literary studies typically require very large amounts of data on which conceptually simple but exceedingly tedious tasks have to be performed. This is a class of situations to which computers are supposed to be ideally suited. A computer can look words up in a dictionary, make word lists and concordances, collect statistics, plot graphs, search for examples, and even perform some kinds of grammatical analysis. But what nobody tells the aspiring computer user is that these machines can all too easily replace one kind of drudgery by another.

Every word of the text must be typed or keypunched on a machine with a pitifully inadequate budget of characters. More or less unsuitable codes must be found for every

---

*Any views expressed in this paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff. This paper was prepared for publication in Computers and the Humanities.

punctuation mark, diacritic, and significant change of type style. The encoding scheme all too easily becomes top-heavy, and in any case it makes such everyday operations as scanning a stream of text, sorting, and merging far more complicated than they would otherwise be. So the researcher omits as many distinctions as he can. He decides that although it would sometimes be convenient to recognize the difference between upper and lower-case letters or between letters with and without accents, these distinctions are really luxuries and should be sacrificed in the interests of economy. Later they turn out to be more important than he thought, but two-thirds of the material has already been keypunched, and it would be unthinkable to do it again.

There are also many hard lessons to be learnt about computer programming. A colleague down the hall has a program which will do almost exactly what is needed, and he will be only too glad to share it. But it turns out to have been written by a student who has since gone elsewhere, or to run on an old or inaccessible computer, or to be incapable, for some more or less obscure reason, of modification in the required way. So a new program has to be written after all, and this takes incomparably longer than anyone, including the programmer, had thought possible.

These difficulties are not peculiar to language processing. But they are probably more acute there than in most other fields, and for reasons that are not difficult to understand. Computers have always been designed primarily to process numeric data. Letters and punctuation marks are useful in programming languages and for printing column headings, and these are the principal reasons for which they

have been provided on key punches and printers. Programmers and other specialists in the field are conditioned to regard computers primarily as numeric calculators which can occasionally, by subterfuge and artifice, be made to do other kinds of work. But language processing is, in fact, a special art which has already accumulated a considerable body of experience and technique. It is unfortunate that programmers who know anything of this are rare. As a result, it is typically the case that both the programmer and his employer imagine that they are doing something new, original, and courageous when in fact they are trying to reinvent the wheel.

The situation is aggravated by the fact that literary and linguistic data-processing tasks are usually large, at least by the standards of university computing centers. A novel is a lot of text, and sooner or later it will probably have to be compared with other novels by the same author or with a larger selection of works from the same period. Fortunately, the time is rapidly approaching when the keypunching of this initial data will be an investment the individual researcher will rarely have to make. A great deal of material is already available, from one source or another, in a form machines can read. In a few years every printing house which wishes to remain competitive will produce a machine-readable version of a text as a natural by-product of the printing process, and it is to be hoped that a systematic effort will be made to insure that this material is not destroyed as it usually is today. What, then, can be done to make this data available to linguists and literary scholars and to enable them to profit as they should from the computer facilities

that are so rapidly becoming cheaper and more powerful?

Computing took on an entirely new complexion for the scientific community as a whole with the introduction of high-level programming languages such as FORTRAN. I shall claim that what is needed for language processing is not so much new program languages, though these are important, but standardization of data formats and a point of view about the different forms that data can take on in and outside the computer.

To a human being a text in English is a text in English. If he is concerned with the substance, he hardly notices what style of type it is printed in, whether titles are in upper case and centered or in upper and lower case and flush left, whether footnotes appear as superscripts or as numbers enclosed in parentheses, and so on. But, in computing, these are crucial matters which determine whether a program will or will not be able to process the data. Furthermore, since the machine works not with hard copy but with an essentially arbitrary code, the possible range of differences in the ways texts are represented for it can be even greater than for the human. What is wanted is a standard code in which any text received from an outside source can be assumed to be.

The trouble about standards is that they are a great deal more attractive to the receiver than to the giver. If the standard is widely adopted, then any program that accepts input in this form will be widely applicable, and the owner of the program will be in a position to receive data from sources still unknown to him when he wrote the program. But the person preparing new data is committed to a set of conventions which, because of their very generality, seem altogether too complicated for his simple purposes. Why should

he have to keep inserting flags to show that the material is in the Greek alphabet when all of his material is in the Greek alphabet, so that the information is completely redundant? Why should he distinguish upper and lower case when he is sure he will never have need of this information? Why should he have to sacrifice the codes for acute and grave accents when his material is all in English and, since there are no accents, he could use these codes for something else.

The answer is that any suggestion that adopting a standard encoding scheme would impose unnecessary restrictions of these kinds is based on altogether too naive a view of the proper place of a standard encoding scheme in linguistic computing as a whole. In fact, the adoption of such a format would, if anything, make the typing and keypunching job easier. Machine formats are for the convenience of machines, and it is a gross and altogether unnecessary imposition ever to require human beings to read or write in them. A typist or keypunch operator frequently needs speecial conventions in order to make up for the small size of the keyboard. But the conventions, as long as they are consistent, should be designed solely with his convenience in mind. Lower case symbols should be chosen for characters which occur frequently and strings of symbols used only for rare characters. But the frequency of characters depends on the particular text, and convenience depends in the last resort on the operator. A new job will typically require a new set of conventions.

The material as initially typed or keypunched will then be converted by a computer program into the standard encoding scheme. Regardless of the standard chosen and of the

conventions used in the input, this conversion process can hardly be anything but trivial. Furthermore, it is often possible to use standard programs for this task -- programs which are capable of working from a simple formalized description of the input conventions that have been used. The amount of computer time spent in the conversion process is bound to be very small in relation to the total amount of computing to be done with the material and, what is more important, in relation to the cost of having large amounts of original material typed or keypunched with rebarbative conventions.

The user should be as free to use his own conventions at output as he is at input. Certainly he should not have to put up with having his material printed in a standardized format that he had no part in the design of and which, for all its good properties, may be entirely inappropriate to his purpose. Furthermore, it is important that linguists and literary scholars should be able to benefit as fully as possible from the wide range of printing devices that are becoming available. Until recently, the device on which hard copy from the computer was produced usually had as restricted a character set as the keypunch. But line printers with 120 or 240 characters are now relatively common. Photographic devices with even larger character sets are available in many places, and photo-composing and other typesetting machines capable of accepting computer output are in regular commercial use. The research worker should be free to have the results of pilot projects and test runs printed on a local machine, compensating for the size of the character set with suitable conventions, and to have more permanent results printed with other conventions on a more adequate,

if more costly, device. This freedom can be purchased at the expense of a few very simple computer routines to convert from one format to another, and, if one of the formats is the common standard, then to that extent the routines themselves will be standard.

That this view of the role of encoding conventions can be made to work, and that it does in fact facilitate the interchange of data and programs among independent workers has been clearly demonstrated. A method of encoding linguistic and other material on magnetic tape known as the "Text and Catalogue System" was designed in 1965 by The RAND Corporation in collaboration with the Centre d'Etudes pour la Traduction Automatique in Grenoble, France, and has since been adopted in several other places. It has made possible a lively exchange of programs and data among the various groups. A complete description of the text encoding scheme, the magnetic tape formats, and some standard programs for manipulating them have been published elsewhere.[1]

It makes sense to talk of standard formats and a standard text encoding scheme only when it can be assumed that the basic medium on which the material is to be recorded is itself standard. If the magnetic tapes used at two installations are fundamentally incompatible so that there is no way of reading at one of them a tape which was written at the other, and any similarities there may be between the

---

[1] See M. Kay and T. Ziehe, Natural Language in Computer Form, The Rand Corporation, RM-4390-PR, February 1965; M. Kay and T. Ziehe, "The Catalog: A Flexible Data Structure for Magnetic Tape," Proc. FJCC, 1965, Washington, D.C. 1965, and M. Kay, F. Valadez and T. Ziehe, The Catalog Input/Output System, The RAND Corporation, RM-4540-PR, March 1966.

encoding conventions used at the two installations are purely of academic interest. The Franco-American proposal just referred to was designed with seven-channel (six plus parity) IBM compatible magnetic tape in mind as the principal medium. Nowadays, there is coming to be greater emphasis on large-capacity/random-access devices for the long-term storage of information. But magnetic tape will doubtless continue for a long time to be the principal medium for storing permanent archives and for transmitting information from one center to another. It is cheap, virtually limitless in capacity, and easy to move from one place to another. But it is now no longer reasonable to look upon the seven-channel magnetic tape as standard, even within the United States. IBM's System 360 uses nine-channel (eight plus parity) tape, and it would be uneconomical as well as inelegant to force the existing format and encoding scheme into this mould. The right thing to do is to devise a format which is appropriate to the new medium, but related in a systematic way to the old one so that conversion from one to another will always be straightforward.

With the old machines, it was obviously appropriate to encode information in units of six binary digits. With the new machines eight binary digits is just as obviously the correct choice. This means that information will now be encoded with a basic alphabet of 256 instead of 64 symbols, so that the assignment of code symbols to actual characters must clearly be different. But basic encoding strategy remains unchanged. Each graphic mark capable of appearing in a text is assigned either to an alphabet, in such a way that characters that tend to occur together in a text will

belong to the same alphabet, or to the set of global codes. Thus, the Roman letters of everyday English text belong to one alphabet and Greek letters to another, since Roman letters tend to occur with other Roman letters and Greek with other Greek letters.

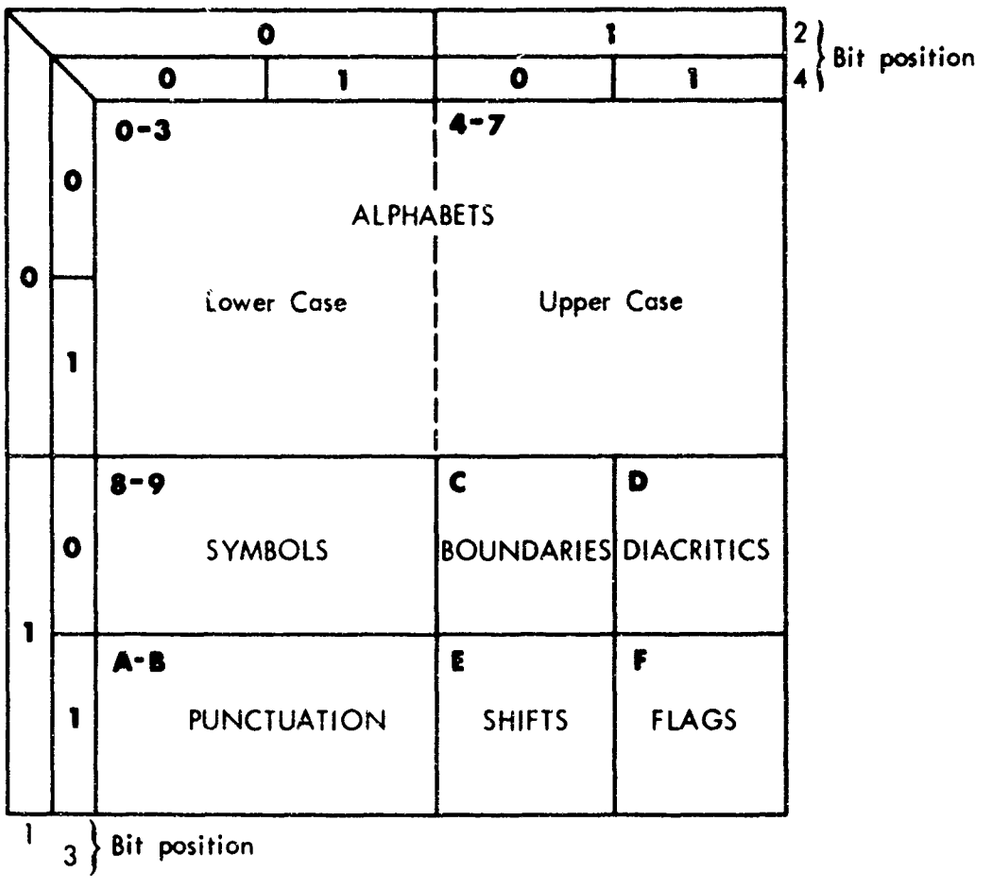
To each alphabet there corresponds a flag, a unique code serving to introduce a string of characters in that alphabet which continues either to the end of the text or to the next flag. Forty-eight of the sixty-four codes in the six-bit encoding scheme are used to represent the members of different alphabets, the other sixteen codes being used for flags and for space character and the like. One of the forty-eight might represent the letter a following the Roman alphabet flag, the letter _ following the Greek alphabet flag, an open quotation mark following the punctuation flag, and so forth. With 256 basic codes, it becomes possible to increase the size not only of the alphabets themselves but of the set of global symbols -- symbols whose interpretation does not depend upon some preceding alphabet flag. In particular, punctuation marks, which are equally common in almost every language, can become global symbols as can diacritics which, in principle at least, can occur with the characters of any alphabet.

The text encoding scheme provides for alphabet flags, substantive characters, and a third set of symbols known as shifts. These serve to distinguish different forms or styles of essentially the same characters. For example, strings of italic or boldface characters are preceded by an appropriate shift code and followed by a code known as a shift terminator. In the six-bit system, each alphabet contains the shifts appropriate to it. But in the eight-

bit system it is economical to treat shifts as flobal symbols.

The six-bit system has a boundary alphabet and another known simply as the symbol alphabet. The boundary alphabet contains codes for separating sentences and paragraphs, syllables and morphemes, stanzas and metric feet, and so forth. The symbol alphabet contains the numerals and other signs used in arithmetic together with such marks as the percent sign, the ampersand, the asterisk, etc. These all become global symbols in the eight-bit system.

The diagram shows how the 256 codes in the eight-bit system are allotted. It is convenient to use the so-called hexadecimal notation to represent an eight-bit code. Each code is represented by a pair of characters, each of which is either a numeral or one of the letters A through F. The first character represents the first four bits, and the second character, the last four. If the letters A through F are thought of as representing the numbers 10 through 15, then there is a single character for every number if the range 0 through 15, and this is the highest number that can be represented with four bits. Each character simply represents the binary equivalent of the number corresponding to it. Thus, the character "9" has the binary equivalent "1001". The letter C corresponds to the number 12, and the binary equivalent of this is "1100". The letters "9C" will represent the eight-bit code "10011100". The characters in the top left-hand corner of the box in the diagram are the ones that can occur first in the hexadecimal representation of codes of the kind shown in that box. Thus,



Outline of the 8-bit text encoding scheme

the first character in the hexadecimal representation of a flag is always "F", and the first four bits of the flag are therefore always "1111". If the first character in the hexadecimal representation of the code is a digit in the range 0 through 7, then the code represents a letter in some alphabet. Codes beginning with "A" or "B" represent punctuation marks, and so on.

As the diagram clearly shows, half of the codes are available for representing characters in the various alphabets. This is enough to allow separate codes for upper and lower case letters. In the six-bit system this distinction was made by means of a shift code. One hundred and twenty-eight codes is also sufficient to represent the characters in most syllabaries. Enough flags are available to distinguish fifteen different alphabets.

The text and catalogue system contains the facilities for encoding natural text that we have outlined. What now is the force of the term "catalogue"? A complete file of textual or other material which has been given a certain kinds of very general hierarchical structure and written in a certain format is called a catalogue.

A catalogue is a set of items known as data which are arranged in the form of a tree. Some data may consist of pieces of text, typically of between one word and one line in length; others may contains numbers representing frequencies or dates or whatever, and others may contain labels which give structure to the file as a whole and make it easier to identify particular data when they are required. Each catalogue is organized in accordance with a map which the person who is assembling the file

must design. The map gives a name to each of the data classes that will be used in the catalogue and says for each what type of encoding will be used for data of that class. In particular, it identifies those classes containing textual material encoded using the scheme we have discussed. In the eight-bit system the map contains, for each data class containing textual material, the alphabet flag under which the majority of the material in the class will be written. If it is English text, then the Roman alphabet flag will be used; if Greek, then Greek, and so on. In the six-bit system each datum in such a class would have to have an alphabet flag as its first code, but in the eight-bit system this is no longer necessary. But if the first code is alphabetic, then it will be interpreted according to the alphabet flag associated with the data class in the map of the catalogue.

The map also shows how the various data classes in a catalogue will be related to one another in the hierarchy. Suppose the catalogue system is used to store a catalogue in the more usual sense -- the catalogue of a library. There will probably be a data class for the principal identifying number -- accession number or class mark -- of a book. Below this in the hierarchy will come the author's name, the title, publisher, date of publication, etc. There will be a data class for each of these categories of information. The map may specify, for example, that author's name and title both come below the main identifying number and that the author's name must come first if there is one. If a book has no author, or the author is unknown, then there will simply be no representative of that class for that

book. In the same way, if a book has two or more authors, then there will be several representatives of the class, and they will all precede the datum of the title class. The catalogue may contain information on the institutional affiliation of an author in which case this will probably be recorded in a class below the author class. Once again, a given author may be shown with no affiliation or with multiple affiliation.

There is very little variation in the physical, as opposed to the logical or rhetorical structure of texts. They come in books, chapters, paragraphs and lines, or in some similar set of size units. It is therefore possible, and very convenient, to standardize the way texts are entered in the catalog system itself. A set of size units has been set up with neutral names so that the same unit can be used for the chapters of a book here, the sections of a report there, and the cantos of a poem somewhere else. The units, in order of decreasing size are corpus, division, section, and entry. There is a data class for each of these which contains identifying labels for particular units of text. There is a second data class for each size unit in which the owner of a file can record comments and annotations to the main text. An entry of text is a unit of about a line, or verse, in length placed beneath a text entry label in the hierarchy. This is the only data class in which actual textual material is recorded, and it is the class to which the vast majority of the data in a text catalog will belong.

A basic set of programs called the Catalog Input/Output System enables a programmer, writing in any language,

to establish a map and enter data into the catalog, and to read data out of a catalog. There are programs for sorting and merging information stored in catalog format and for rewriting the information from one catalogue in another, but with a different map. Some special programs exist for handling text on a word-by-word or sentence-by-sentence basis. A program has been written which will read a text catalog, remove prefixes and suffixes, separate compound words, look each segment up in a dictionary, and write a new catalogue in which each segment is presented with grammatical, semantic or other information from the dictionary. Most of these programs are written in a high-level programming language so that, as far as possible, they are easy to understand and modify. The more competent individuals and groups adopt the text and catalog system, the wider the available range of programs will become.